# Arrays

## Table of Contents



## What are arrays

An array is a single variable that can hold more than one value. An array is a special kind of variable that contains multiple values. If you think of a variable as a container that contains a value, an array can be thought of as a container with compartments where each compartment is able to store an individual value.

Arrays let you work with a large amount of similar data. Arrays can be of any length and it is easy to manipulate all values at once. Each value within an array is called an element. An array element is accessed via its index.

## Creating an array

To create an array in PHP, use square brackets [ and ] containing the values you want to store, separated by commas:

```
$daysOfTheWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
```

Arrays in PHP can also be defined using the array keyword. The following code is equivalent to the square bracket notation above:

```
$daysOfTheWeek = array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday");
```

The above examples create an indexed array. Each array element can be accessed using its index. For example:

```
$daysOfTheWeek[0]
```

Would return the value “Monday”

If you want to create an associative array, where each element is identified by a string index (or a key) rather than a number, you need to use the => operator, as follows:

```
$course = ["title" => "Web Development", "code" => "ICA50611", "version" => 2];
```

or

```
$course = array("title" => "Web Development", "code" => "ICA50611", "version" => 2);
```

This creates an array with three elements: “Web Development”, which has an index of “title“; “ICA50611”, which has an index of “code”; and 2 , which has an index of “version”.

To access the array elements you can use the syntax:

```
echo $course["title"];
```

### Activity 1. Creating arrays

Create the following PHP file:

```
1  <pre>
2  <?php
3    $daysOfTheWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
```

```
4
5     for ($i=0; $i<5; $i++)
6     {
7        echo $daysOfTheWeek[$i] . "\n";
8     }
9
10    $course = ["title" => "Web Development", "code" => "ICA50611", "version" => 2];
11
12    echo $course["title"];
13 ?>
14 </pre>
```

*Additional practice*

1. Use the above example to display the code and version of the $course array.
2. Store months of the year in an array, loop through the array to display the months.

## Changing and adding elements

You can change or add array elements by using the index of the array element. For example:

```
$daysOfTheWeek[1] = “Tue”;
```

Will change Tuesday to Tue

```
$daysOfTheWeek[5] = “Saturday”;
```

Will add Saturday to the end of the array.

You can also use:

```
$daysOfTheWeek[] = “Saturday”;
```

The above line of code would also add Saturday to the end of the array. PHP will automatically assign the next available index.

You can also create an empty array:

```
$daysOfTheWeek = array();
```

Changing elements of associative arrays works the same way:

```
$course[“title”] = “Web Design”;
```

# Outputting an entire array with print_r()

You can use print_r() to display the whole array:

```
print_r($daysOfTheWeek);
```

If you’d rather store the output of print_r() in a string, rather than displaying it, you can pass a second true argument to the function:

```
$arrayStructure = print_r( $array, true );
```

### Activity 2. Changing arrays

Modify the $daysOfTheWeek array to include Saturday and Sunday, display the array using print_r()

# Counting array elements

You can use the count() function to count the number of elements in an array.

### Activity 3. Counting array elements

Modify the $daysOfTheWeek example so that it now uses the count() function:

```
1  $daysOfTheWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
2
3     $daysOfTheWeek[5] = "Saturday";
4     $daysOfTheWeek[] = "Sunday";
5
6     for ($i=0; $i<count($daysOfTheWeek); $i++)
```

```
7    {
8       echo $daysOfTheWeek[$i] . "\n";
9    }
```

You can also use the count() function with associative arrays:

```
$course = ["title" => "Web Development", "code" => "ICA50611", "version" => 2];

echo count($course);
```

# Using foreach to loop through an array

The foreach statement is a special kind of looping statement that works only on arrays. You can use it in two ways. You can either retrieve just the value of each element, or you can retrieve the element's key and value.

## Activity 4. Foreach

Modify the exercise so you are now using the foreach statement to loop through the array:

```
1    $daysOfTheWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
2
3    $daysOfTheWeek[5] = "Saturday";
4    $daysOfTheWeek[] = "Sunday";
5
6    foreach ($daysOfTheWeek as $value)
7    {
8       echo $value ."\n";
9    }
```

The foreach loop continues to iterate until it has retrieved all the values in the array, from the first element to the last. On each pass through the loop, the $value variable gets set to the value of the current element.

# Using the foreach to loop through keys and values

The foreach statement can be used to loop through keys and values:

```
$course = ["title" => "Web Development", "code" => "ICA50611", "version" => 2];

foreach ($course as $key => $value)
{
   echo "$key = $value\n";
}
```

## Activity 5. Foreach keys and values

Try the above code.

## Alternative syntax

In my examples I've used the <pre> </pre> tags outside of the PHP code so that the spacing and line breaks are kept. I've included a \n (backslash n) to include a line break at each loop iteration. Often you will want to format your output using HTML and CSS styling. Mixing up the HTML with the PHP can result in code that is hard to read. PHP features an alternative syntax for conditional and looping structures: if, for, while, foreach.

It differs from standard syntax of control (if, for, while, foreach) in that you can type colons ( : ) instead of opening braces and special key words (endif, endfor, endwhile, endforeach) instead of the closing ones.

### *If statement example*

```
if (1 > 0) :
  echo "Hello";
endif;
```

*While loop example*

```
$i=1;
while ($i<10) :
  echo $i;
  $i++;
endwhile;
```

*foreach loop example*

```
foreach ($daysOfTheWeek as $value) :
?>
   <li><?=$value?></li>

<?php endforeach; ?>
```

### Activity 6. Alternative syntax

```
1  <h1>Exercise</h1>
2  <ul>
3  <?php
4     $daysOfTheWeek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
5
6     $daysOfTheWeek[5] = "Saturday";
7     $daysOfTheWeek[] = "Sunday";
8
9     foreach ($daysOfTheWeek as $value) :
10 ?>
11    <li><?=$value?></li>
12
13 <?php endforeach; ?>
14 </ul>
```

## Exercises

1. Create an array called streams add the streams names you are currently studying. Display each of the stream name one per line
2. Modify the above exercise so the stream names are displayed in a bullet list, try the exercise using the alternative foreach syntax.
3. Create an array with all the student names in the class. Display the names one per line.